Docket No : **POU920000119US1**

Inventor : **King-Smith, et al.**
Title : **DYNAMICALLY OPTIMIZING THE TUNING OF SOCKETS ACROSS INDETERMINATE ENVIRONMENTS**

APPLICATION FOR UNITED STATES

LETTERS PATENT

"Express Mail" Mailing Label No.: **EK830427681US**
Date of Deposit: **June 27, 2001**

I hereby certify that this paper is being deposited with the United States Postal Service as "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to: Box Patent Application, Assistant Commissioner for Patents, Washington, D.C. 20231.

Name: **Ann S. Lund**

Signature: Ann S. Lund

INTERNATIONAL BUSINESS MACHINES CORPORATION

DYNAMICALLY OPTIMIZING THE TUNING OF SOCKETS ACROSS INDETERMINATE ENVIRONMENTS

Technical Field

[0001] This invention relates, in general, to optimizing system resources, and in particular, to dynamically tuning sockets across an indeterminate number of socket connections and unknown network types.

Background of the Invention

[0002] In computing environments that can be dynamically changed, such as in parallel and cluster environments, optimal tuning of system resources is a challenge and tedious task. In particular, it is very difficult to tune the system resources for optimal performance when, for instance, the number of nodes and thus, the number of sockets, is variable, or when the type of network is unknown.

[0003] Thus, previously, a single set of resources has been tuned and shared across several applications. Alternatively, parameters of the resources have been adjusted with severe impact to the system. That is, the system would be stopped, adjustments would be made, and then the system would be started again, thus impacting other applications on the system.

[0004] Based on the foregoing, a need still exists for a capability that enables system resources to be optimally set for dynamic computing environments. For example, a need exists for a capability that enables the tuning of socket parameters for indeterminate environments.

Summary of the Invention

[0005] The shortcomings of the prior art are overcome and additional advantages are provided through the provision of a method of tuning sockets of a computing environment. The method includes, for instance, dynamically determining information relating to a current configuration of the computing environment; and setting one or more parameters of a socket of the computing environment based on the dynamically determined information.

[0006] In a further embodiment, a method of tuning sockets of a computing environment is provided. The method includes, for instance, determining, in response to opening a socket of the computing environment, information relating to a current configuration of the computing environment, the information including at least one of information relating to a network of the computing environment coupled to the socket and information relating to the socket; and setting one or more parameters of the socket based on the determined information, wherein the one or more parameters reflect the current configuration of the computing environment.

[0007] System and computer program products corresponding to the above-summarized methods are also described and claimed herein.

[0008] Advantageously, the capabilities of the present invention enable the tuning of system resources in changing environments. For example, socket parameters are dynamically set based on the current configuration of the environment (e.g., based on the current number of socket connections and/or the network type). This tuning is performed without stopping the system and with no need for static allocations.

[0009] In one example, the capabilities of the present invention provide optimized values to the Transmission Control Protocol (TCP)/Internet Protocol (IP) socket that utilizes the advertised window feature of TCP/IP to prevent exhaustion of system network resources and poor performance.

[0010] Additional features and advantages are realized through the techniques of the present invention. Other embodiments and aspects of the invention are described in detail herein and are considered a part of the claimed invention.

Brief Description of the Drawings

[0011] The subject matter which is regarded as the invention is particularly pointed out and distinctly claimed in the claims at the conclusion of the specification. The

foregoing and other objects, features, and advantages of the invention are apparent from the following detailed description taken in conjunction with the accompanying drawings in which:

[0012] FIG. 1a depicts one embodiment of a computing environment incorporating and using aspects of the present invention;

[0013] FIG. 1b depicts one example of various buffers of the computing environment of FIG. 1a, one or more of which are tuned in accordance with an aspect of the present invention;

[0014] FIG. 2 depicts one example of information used by one aspect of the present invention to tune one or more of the socket buffers of FIG. 1b;

[0015] FIG. 3 depicts one embodiment of the logic used by a home node to tune one or more of its socket buffers, in accordance with an aspect of the present invention; and

[0016] FIG. 4 depicts one embodiment of the logic used by a remote node to tune one or more of its socket buffers, in accordance with an aspect of the present invention.

Best Mode for Carrying Out the Invention

[0017] In accordance with an aspect of the present invention, sockets are dynamically optimized for indeterminate environments. For example, sockets are optimized for changing environments (i.e., varying number of socket connections and/or different network types). To dynamically optimize a particular socket, application specific information available at the time the socket is opened and/or information available from the socket itself is utilized. This information is used to determine the optimal socket tuning needed for efficient internet protocol (IP) traffic between the node initializing the socket and the node at the other end (i.e., the remote node) of the socket connection.

[0018] One embodiment of a computing environment incorporating and using aspects of the present invention is described with reference to FIG. 1a. A computing environment 100 includes, for instance, a plurality of nodes 102 coupled to one another via one or more networks 104. Although two nodes and one network are illustrated, it is well known that a computing environment can include many more nodes and networks. For example, a particular node can be coupled to one or more other nodes via one network type, and further coupled to one or more other nodes via another network type, etc.

[0019] A node 102 includes an operating system 106, such as AIX. Operating system 106 includes, for instance, at

least one device driver 108 and at least one socket 110, such as a TCP/IP socket. The number of device drivers is based, for instance, on the number of adapters in the environment; and the number of sockets is based, for example, on the desired number of socket connections between the node and one or more other remote nodes.

[0020] Coupled to operating system 106 are one or more network adapters 112. Each network adapter is used to couple its associated node to a particular network, which corresponds to that adapter type. Network 104 includes any IP capable network, such as the Ethernet, Token Ring, FDDI, ATM, SP Switch, Fiber Channel, etc.

[0021] Further details regarding socket 110 and adapter 112 are described with reference to FIG. 1b. In one embodiment, socket 110 includes at least one send buffer 120 used to send data from the socket, and at least one receive buffer 122 used to receive data at the socket. Additionally, adapter 112 includes, for instance, at least one transmit buffer 124 used to transmit data across the network.

[0022] The buffers are used to communicate between the nodes. For efficient communication, however, various of the buffers are tuned appropriately. For example, the send and receive buffers of a socket are tuned. This tuning includes, for instance, establishing optimal settings for the parameters of the socket buffers (i.e., socket parameters), even though the environment includes an

indeterminate number of socket connections and/or unknown network types. That is, optimal settings are determined even though the environment is changing.

[0023] The setting of the socket parameters utilizes certain information that is dynamically determined, when for instance, the node opens the socket. This information is available from the application opening the socket and/or from the socket itself. One example of the information used in tuning the parameters is described with reference to FIG. 2.

[0024] In one embodiment, the information includes, for instance:

(a) The number of remote sockets 200 to be opened. This information is available from the application requesting the socket connection (i.e., on the home node). This application determines at socket open time the total number of remote sockets.

[0025] The following information, which is extracted by the node opening the socket, is extracted from the opened socket itself:

(b) A network adapter maximum segment size 202, which indicates the size of the largest message that can be sent for the particular network being used. In TCP/IP, this

information is retrieved as the TCP_MAXSEG option from the IPPROTO_TCP level;

(c) A current socket send buffer size 204, which indicates how much data the socket is going to attempt to send. In TCP/IP, this information is retrieved as the SO_SNDBUF option from the SOL_SOCKET level;

(d) A current socket receive buffer size 206, which indicates how much data the socket can receive. In TCP/IP, this information is retrieved as the SO_RCVBUF option from the SOL_SOCKET level;

(e) A current socket maximum buffer limit (sb_max), which indicates the maximum amount of buffer space that a single socket could obtain to stage data. It is retrieved, for instance, from the sb_max field of the socket parameter structures; and

(f) An adapter transmit limit 210, which indicates either the maximum number of packets or the maximum amount of data that an adapter can handle at any one time. It is obtained using, for instance, ioctl for the adapter.

[0026] Some or all of the information listed above is used to set one or more parameters associated with the socket. For instance, the information, which is based on the current configuration of the computing environment, is used to set the current socket send buffer size and/or the current socket receive buffer size.

Sub C1

[0027] These parameters are set on the node initiating the socket connection (i.e., the home node), as well as on the node at the other end of the connection (i.e., the remote node). One embodiment of the logic associated with performing the tuning by the home node is described with reference to FIG. 3. As one example, the logic runs in the operating system of the home node and is run at the time of opening the socket.

[0028] Referring to FIG. 3, initially, a determination is made as to whether the current socket send buffer size (SO_SNDBUF) is less than a predetermined value, INQUIRY 300. In one example, this predetermined value is equal to four times the network adapter maximum segment size (TCP_MAXSEG). If the retrieved SO_SNDBUF value is less than four times TCP_MAXSEG, then the SO_SNDBUF is set equal to the predetermined value, STEP 302. That is, in this example, SO_SNDBUF is set equal to four times TCP_MAXSEG. This is to ensure minimal data streaming from the socket.

[0029] Subsequently, or if the socket send buffer size is not less than the predetermined value, a further determination is made as to whether the socket send buffer

size is greater than the amount of data that can be transmitted, INQUIRY 304. For instance, a determination is made as to whether the retrieved SO_SNDBUF value is greater than the maximum number of packets in the adapter transmit queue times TCP_MAXSEG, or greater than the adapter transmit buffer size, depending on how the data is being transmitted (e.g., packets or buffers).

[0030] Should the socket send buffer size be greater than the amount of data that can be transmitted, then SO_SNDBUF is set to the maximum number of packets in the adapter transmit buffer times TCP_MAXSEG or it is set to the maximum adapter buffer size to prevent sending more data than the network adapter can handle, STEP 306.

[0031] Thereafter, or if the send buffer value is not greater than the amount of data that can be transmitted, the socket receive buffer size is set, STEP 308. In one example, the socket receive buffer size (SO_RCVBUF) is set to a large size (e.g., 512k) so that each socket can have a reasonable receive buffer to receive the data.

[0032] Subsequent to setting the socket receive buffer size, a further determination is made as to whether the socket send buffer size or the socket receive buffer size is greater than the current socket maximum buffer limit (sb_max), INQUIRY 310. Should the socket send buffer size (SO_SNDBUF) or socket receive buffer size (SO_RCVBUF) exceed sb_max, then the socket maximum buffer limit is used for whichever buffer size is in excess to prevent a system error

message indicating a socket is exceeding the socket maximum buffer limit, STEP 312. This completes the tuning of the send and receive buffers on the home node.

[0033] In addition to tuning the send and receive buffers of the socket on the home node, the send and receive buffers of the socket at the remote node are also tuned. One embodiment of the logic associated with this tuning is described with reference to FIG. 4. As one example, the logic runs in the operating system of the remote node and is run at the time of opening the socket.

[0034] Referring to FIG. 4, initially, the remote node receives from the home node the number of remote sockets to be opened. Then, based on the number of remote sockets to be opened, the current socket send buffer size is set on the remote node using a predefined equation, STEP 400. In one example, the predefined equation includes: (maximum number of packets per adapter of the adapter transmit limit (or maximum buffer of adapter transmit limit ÷ TCP_MAXSEG) ÷ number of remote sockets to be opened (or four if less than four)) × TCP_MAXSEG. This equation is used to determine a maximum amount of data that can be sent by the socket based on the current configuration.

[0035] Thereafter, a determination is made as to whether the socket send buffer size derived from the above equation is greater than the socket maximum buffer limit (sb_max), INQUIRY 402. If so, then sb_max is used for the socket send

buffer size, instead of the derived socket send buffer size, STEP 404.

[0036] Next, or if the socket send buffer size is not greater than sb_max, then a further determination is made as to whether the socket receive buffer size is less than a predetermined value, INQUIRY 406. In one example, this predetermined value is equal to four times the network adapter maximum segment size (TCP_MAXSEG). If the socket receive buffer value is less than four times TCP_MAXSEG, then SO_RCVBUF is set equal to the predetermined value, STEP 408. This is to ensure minimal data streaming from the TCP/IP socket.

[0037] Subsequently, or if the socket receive buffer size is not less than the predetermined value, then a further determination is made as to the whether the socket receive buffer size is greater than the amount of data that the adapter can receive, INQUIRY 410. Should the socket receive buffer size be greater than the amount of data that the adapter can receive, then the socket receive buffer size is set, STEP 412. In one embodiment, it is set equal to the (maximum number of packets per adapter of the adapter transmit limit (or maximum buffer of the adapter transmit limit divided by TCP_MAXSEG) divided by the number of remote TCP/IP sockets to be opened (or four if less than four)) × TCP_MAXSEG.

[0038] Described in detail above is a capability for optimally tuning a socket. The capability utilizes

application specific data and/or information about the socket connection available from the socket to set one or more parameters of the socket. Each socket is optimized, so that the amount of data sent by one or more sockets of a particular network does not exceed the capabilities of the network adapters.

[0039] For the home node sending out data, the tuning parameters ensure that a minimal number of IP packets will be sent per socket, thus providing some level of data streaming. Further, it ensures that the number of packets will not exceed the adapters ability for handling packets. The remote node ensures that enough space is allocated, so that it can receive the full amount of data that the home node can send.

[0040] For the remote node, each connection is provided a portion of the total amount of space that the home node can receive. This prevents the remote nodes from flooding the home node with more data than it can handle. This uses the advertised windowing ability of TCP/IP across the remote connections to limit the amount of data sent per TCP/IP socket.

[0041] Advantageously, if there are multiple types of networks available, the capability of the present invention automatically adjusts the buffer allocations based on the segment size of the network selected per socket connection. If several parallel or cluster jobs are run at different numbers of sockets, then the technique adjusts the

allocations of the buffers to optimize to the number of sockets. This is especially advantageous in cluster environments where the number of sockets and/or number of nodes available can be dynamic, which has to be accounted for each time a job is started.

[0042] Although aspects of the invention are described with reference to TCP/IP, the invention is not limited to such an environment. For example, the techniques can be applied to any implementation of IP, including, but not limited to, UDP/IP.

[0043] The present invention can be included in an article of manufacture (e.g., one or more computer program products) having, for instance, computer usable media. The media has embodied therein, for instance, computer readable program code means for providing and facilitating the capabilities of the present invention. The article of manufacture can be included as a part of a computer system or sold separately.

[0044] Additionally, at least one program storage device readable by a machine, tangibly embodying at least one program of instructions executable by the machine to perform the capabilities of the present invention can be provided.

[0045] The flow diagrams depicted herein are just examples. There may be many variations to these diagrams or the steps (or operations) described therein without departing from the spirit of the invention. For instance,

the steps may be performed in a differing order, or steps may be added, deleted or modified. All of these variations are considered a part of the claimed invention.

[0046] Although preferred embodiments have been depicted and described in detail herein, it will be apparent to those skilled in the relevant art that various modifications, additions, substitutions and the like can be made without departing from the spirit of the invention and these are therefore considered to be within the scope of the invention as defined in the following claims.